Mapping and Markdown Tutorial

Nicholas Nagle

9/5/2019

## Learning Objectives

* Practice a little bit of tidyverse and ggplot
* Create your first R markdown document
* Learn how to import and map shapefiles in R.

## R Markdown helps

* <https://rmarkdown.rstudio.com/authoring_basics.html>
* <https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>

## 

The following R chunk loads the libraries that I will use. I tend to put them at the top of the markdown file so that other people can immediately see what the requirements are.

* tidyverse: I always use tidyverse
* sf: For working with shapefiles
* ggspatial: useful for plotting spatial data: includes openstreetmap basemap

library(tidyverse)

## ── Attaching packages ──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.1.1 ✔ purrr 0.3.2   
## ✔ tibble 2.1.3 ✔ dplyr 0.8.3   
## ✔ tidyr 0.8.3.9000 ✔ stringr 1.4.0   
## ✔ readr 1.3.1 ✔ forcats 0.4.0

## ── Conflicts ─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(sf)

## Linking to GEOS 3.6.2, GDAL 2.2.3, PROJ 4.9.3

library(ggspatial)

## Load the data

Most data in geographic research seem to come in Microsoft Excel files (.xls), comma separated filed (.csv) or shapefiles (.shp). For microsoft excel files, I find it easiest to save them as comma separated files in excel first.

For csv and shp files, the way to load them into R is basically the same. It should like something like this: any\_name\_you\_want <- read\_csv('file.csv') or any\_name\_you\_want <- st\_read('file.shp'). The read\_csv function is in the readr library (which is part of the tidyverse), and the st\_read function is part of the sf library (which is not part of the tidyverse). There are packages for reading Excel files, but I recommend saving as csv first.

Tip: When you type a directory and file path in R, you can use the tab button, and it will show you options to help complete the filename. Use this to avoid misspelling file names and folders.

The data for this lab are a shapefile of census tracts for Knox County, with columns for total population, population between ages 18 and 21, median household income and median house price.

Load the shapefile ‘knox\_acs.shp’ into R. I gave it the name ‘knox’. This assumes that you are running this file in the same place where you saved the shapefile. When the st\_read function loads a shapefile, the R object it creates is called a sf or “simple feature”. It is a dataframe with extra information about the spatial coordinates.

knox <- st\_read('knox\_acs.shp')

## Reading layer `knox\_acs' from data source `/home/nnagle/Dropbox/classes/Geog415/geog415\_f19/labs/lab02\_rmarkdown/knox\_acs.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 112 features and 6 fields  
## geometry type: POLYGON  
## dimension: XY  
## bbox: xmin: -84.27347 ymin: 35.79381 xmax: -83.65092 ymax: 36.18648  
## epsg (SRID): 4269  
## proj4string: +proj=longlat +datum=NAD83 +no\_defs

Click on the object knox in in the Environment window in the top right. It will open up the data in the top left. You will see the data columns. There is also a columns called “geometry”. This column has the information necessary for mapping the data. You almost never want to change this column.

## Create a choropleth map of the Median Household Population.

Creating a choropleth is easy with ggplot. Technically, a choropleth is a polygon geometry, but there is a special function geom\_sf for mapping with simple features. The most important aesthetic is the fill color. The color aesthetic is the border color of each polygon. It’s probably not what you want.

ggplot(knox) +   
 geom\_sf(aes(fill=MEDHHINC)) +   
 labs(fill = "Median Household Income")



## Removing the lat/long coordinates

Most people aren’t familiar enough to have the lat/long coordinates be meaningful, so they should be removed. I couldn’t remember how to do that, but Googling “geom\_sf” and “turn off axis labels” brought me to [this page](https://stackoverflow.com/questions/49836184/cant-remove-gridlines-when-plotting-with-geom-sf) which showed me the coord\_sf(datum=NA) function.

ggplot(knox) +   
 geom\_sf(aes(fill=MEDHHINC)) +   
 labs(fill = "Median Household Income") +  
 coord\_sf(datum=NA)



## Changing the color ramp

The default color ramp is pretty lousy. I recommend always sticking to color palettes from [ColorBrewer](http://colorbrewer2.org/#type=sequential&scheme=Blues&n=3). They’ve been carefully chosen to have color that most people can easily distinguish.

The way you change the scaling of any aesthetic in R is with a scale function. The function for changing the fill aesthetic using color brewer is scale\_fill\_distiller. The following chunk shows this.

Ideally, you’d turn the continuous variable into discrete categories before drawing choropleth, but we’ll save that for a later lab.

Also, notice how I rescaled the variable by 10,000, and how I created a new line in the legend with “”.

ggplot(knox) +   
 geom\_sf(aes(fill=MEDHHINC/10000)) +   
 scale\_fill\_distiller(palette='Blues')+  
 labs(fill = "Median Household Income\n($1000s)") +  
 coord\_sf(datum=NA)



## Putting an open street map under your map

Finally, it might be useful to pub a general purpose map under your map for reference. The annotation\_map\_tile function in the ggspatial package does that.

Play with the alpha level to adjust the transparancy of your thematic map.

Also notice that I want the basemap under my choropleth, so I listed the basemap layer before listing the geom\_sf layer. Layers are drawn in the order in which they appear.

ggplot(knox) +   
 annotation\_map\_tile() +  
 geom\_sf(aes(fill=MEDHHINC/10000), alpha=.7) +   
 scale\_fill\_distiller(palette='Blues')+  
 labs(fill = "Median Household Income\n($1000s)") +  
 coord\_sf(datum=NA)

## Loading required namespace: raster

## Zoom: 9



A message saying “zoom: 9” is shown. It says what zoom level was selected for the basemap. You can safely ignore it. The annotation\_map\_tile() is pretty good at picking an appropriate zoom level.

## Calculating a correlation coefficient in R.

The formula for correlation coefficient is cor(x,y). It will calculate the correlation coeffience between variables x and y and return a single number. If there are missing data, it will return a missing value to. To ignore the missing data, the function is cor(x, y, use = 'complete.obs').

To calculate a correlation coefficient between Median House Value and Median Household Income, I will show how to do that two ways: first not using the tidyverse, and the second with the tidyverse.

### Calculating the correlation coefficient the non-tidyverse way.

The cor function can be used outside the tidyverse. It just needs two columns of data. The way to access a column of data outside the tidyverse is like so: knox$MEDHHINC and knox$MEDHVALUE. The formula is data\_frame$column\_name.

corr\_coef <- cor(knox$MEDHVALUE, knox$MEDHHINC, use='complete.obs')  
corr\_coef # Just type the object to print it out.

## [1] 0.8389087

If you want to use value in a sentence, you would do it like so: “0.8389087”. Example, the correlation coefficient betweenm Median House Value and Median Household Income is 0.8389087.

### Using correlation the tidyverse way

The plan is to summarize the knox dataframe. This will create a new dataframe. I will call the new frame summary\_stat. I will also calculate means and standard deviations while I’m at it.

summary\_stat <- knox %>%  
 summarize(mean\_HHINC = mean(MEDHHINC, na.rm=TRUE),  
 sd\_HHINC = sd(MEDHHINC, na.rm=TRUE),  
 mean\_HVALUE = mean(MEDHVALUE, na.rm=TRUE),  
 sd\_HVALUE = sd(MEDHVALUE, na.rm=TRUE),  
 cor\_HVALUE\_HHINC = cor(MEDHHINC, MEDHVALUE, use='complete.obs'))  
summary\_stat # just the the dataframe to print it out.

## Simple feature collection with 1 feature and 5 fields  
## geometry type: POLYGON  
## dimension: XY  
## bbox: xmin: -84.27347 ymin: 35.79381 xmax: -83.65092 ymax: 36.18648  
## epsg (SRID): 4269  
## proj4string: +proj=longlat +datum=NAD83 +no\_defs  
## mean\_HHINC sd\_HHINC mean\_HVALUE sd\_HVALUE cor\_HVALUE\_HHINC  
## 1 56057.96 26640.32 174410.9 86960.74 0.8389087  
## geometry  
## 1 POLYGON ((-83.69458 35.9438...

In doing that, I saw that it also printed out a bunch of stuff about the map. We can delete that with the df\_spatial function in the ggspatial package. The df\_spatial function just takes a sf object and returns a clean (non-spatial) data frame.

summary\_stat <- knox %>%  
 df\_spatial() %>%  
 summarize(mean\_HHINC = mean(MEDHHINC, na.rm=TRUE),  
 sd\_HHINC = sd(MEDHHINC, na.rm=TRUE),  
 mean\_HVALUE = mean(MEDHVALUE, na.rm=TRUE),  
 sd\_HVALUE = sd(MEDHVALUE, na.rm=TRUE),  
 cor\_HVALUE\_HHINC = cor(MEDHHINC, MEDHVALUE, use='complete.obs'))  
summary\_stat # just type the dataframe to print it out.

## # A tibble: 1 x 5  
## mean\_HHINC sd\_HHINC mean\_HVALUE sd\_HVALUE cor\_HVALUE\_HHINC  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 58244. 25355. 181319. 89796. 0.861

If you want to use value in a sentence, you would do it like so: “0.8612655”. Example, the correlation coefficient betweenm Median House Value and Median Household Income is 0.8612655.